PEMROGRAMAN JARINGAN

“Eksplorasi Mandiri 3”

DOSEN PENGAMPU:

Randi Proska Sandra, M.Sc.

OLEH:

Muhammad Alfarobi

23343011

PROGRAM STUDI INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

A. Latar Belakang Program Aplikasi

Perkembangan teknologi komunikasi menuntut proses pengiriman informasi yang semakin cepat dan otomatis. Dalam konteks praktikum jaringan dan pengembangan aplikasi web, kemampuan untuk mengirim email secara otomatis dari server menjadi salah satu kebutuhan penting, terutama untuk fitur seperti notifikasi, verifikasi akun, pengiriman tagihan, serta laporan sistem. Node.js dipilih sebagai platform utama karena sifatnya yang ringan, cepat, dan memiliki ekosistem library yang luas, termasuk modul nodemailer yang khusus digunakan untuk layanan email.

Pada proses pembuatan aplikasi ini, dibangun sebuah Email Sender sederhana berbasis Node.js dan Express.js. Aplikasi dirancang untuk menerima data melalui REST API, memprosesnya, dan mengirimkan email ke penerima menggunakan akun Gmail yang telah dikonfigurasi. Penggunaan App Password Gmail dipilih karena Google telah menghentikan akses "less secure apps", sehingga pengiriman email wajib menggunakan autentikasi dua langkah dan app password untuk keamanan yang lebih tinggi.

Beberapa kendala teknis muncul selama pengembangan, seperti pengaturan autentikasi Gmail, kesalahan struktur direktori, modul yang tidak ditemukan, hingga variabel yang tidak terdefinisi. Setiap masalah tersebut menjadi bagian dari proses pembelajaran penting dalam memahami struktur proyek Node.js, cara mengelola environment variable, serta debugging ketika aplikasi tidak berjalan sebagaimana mestinya. Dengan menyelesaikan berbagai error tersebut, aplikasi akhirnya dapat berjalan dan mengirim email melalui endpoint API yang telah dibuat.

Pembuatan aplikasi Email Sender ini menjadi pengalaman praktis dalam memahami bagaimana sebuah server Node.js berkomunikasi dengan layanan eksternal, mengelola konfigurasi sensitif seperti kredensial email, serta merancang arsitektur API yang terstruktur. Aplikasi ini juga dapat dikembangkan lebih lanjut menjadi layanan otomatis seperti pengiriman invoice, reset password, notifikasi, atau integrasi sistem lainnya.

B. Langkah-Langkah Pembuatan Aplikasi

1. Buat folder besar nya bernama mailer
2. Ambil package.json dengan perintah npm init -y di terminal

   File package.json berfungsi sebagai identitas dan konfigurasi proyek, yang berisi daftar dependensi, nama aplikasi, versi, serta script untuk menjalankan server. File ini menjadi pusat manajemen library pada aplikasi Node.js.
3. Ambil package-lock.json dengan perintah npm i express nodemon di terminal

   File package-lock.json dibuat otomatis oleh NPM untuk mengunci versi setiap paket yang digunakan sehingga instalasi dependensi menjadi konsisten di semua perangkat. File ini memastikan aplikasi tidak rusak akibat perbedaan versi library.
4. Buat file server.js

   File server.js adalah titik masuk utama aplikasi, tempat Express diinisialisasi, middleware diaktifkan, rute dihubungkan, dan server dijalankan pada port tertentu. File ini memastikan aplikasi siap menerima request dari client.

   Source code:

```
const express = require('express');
const appRoute = require('./routes/route.js')
const app = express();
const PORT = process.env.PORT || 5000;
app.use(express.json());

/** routes */
app.use('/api', appRoute);
app.listen(PORT, () => {
    console.log('Server is running on http://localhost:${PORT}')
```

```
})
```

5. Buat folder routes
6. Buat file route.js

   File routes.js berfungsi sebagai pengatur jalur API, tempat mendefinisikan endpoint seperti /sendmail yang nantinya memanggil fungsi pada controller. Dengan file ini, alur request dari client dapat diarahkan dengan rapi dan terstruktur.

   Source code:

```
const router = require('express').Router();
const { signup, getbill } = require('../controller/appController.js')

/** HTTP Request */
router.post('/user/signup', signup);
router.post('/product/getbill', getbill);
module.exports = router;
```

7. Buat folder controller
8. Buat file appController.js

   File appController.js berfungsi sebagai pusat logika aplikasi, yaitu mengolah request dari client, menyiapkan konfigurasi Nodemailer, membuat struktur email, dan menjalankan proses pengiriman email menggunakan akun Gmail. File ini juga menangani response yang akan dikirim kembali ke client, baik ketika email berhasil dikirim maupun ketika terjadi error.

   Source code:

```
const nodemailer = require('nodemailer');
const Mailgen = require('mailgen');

const { EMAIL, PASSWORD } = require('../env.js')

/** send  mail from testing account */
const signup = async (req, res) => {

    let testAccount = await nodemailer.createTestAccount();

    let transporter = nodemailer.createTransport({
        host: "smtp.ethereal.email",
        port: 587,
        secure: false,
        auth: {
            user: testAccount.user,
            pass: testAccount.pass,
        },
    });

    let message = {
        from: '"Robi" <robi@example.com>',
        to: "bar@example.com, baz@example.com",
        subject: "Hello",
        text: "Succesfully Register with us.",
        html: "<b>Succesfully Register with us.</b>",
        }

    transporter.sendMail(message).then((info) => {
        return res.status(201)
        .json({
            msg: "you should receive an email",
            info : info.messageId,
            preview: nodemailer.getTestMessageUrl(info)
        })
    }).catch(error => {
```

```
        return res.status(500).json({error})
    })

// res.status(201).json("Signup Successfully...!");
}

/** send mail from real gmail account */
const getbill = (req, res) => {

    const { userEmail } = req.body;

    let config = {
        service : 'gmail',
        auth : {
            user: EMAIL,
            pass: PASSWORD
        }
    }

    let transporter = nodemailer.createTransport(config)

    let MailGenerator = new Mailgen({
        theme: "default",
        product : {
            name: "Mailgen",
            link : 'https://mailgen.js/'
        }
    })

    let response = {
        body: {
            name : "Muhammad Alfarobi",
            intro: "Your bill has arrived!",
            table : {
                data : [
                    {
                        item : "Nodemailer Stack Book",
                        descrption: "A Backend application",
                        price : "$10.99",
                    }
                ]
            },
            outro: "Looking for to do more business"
        }
    }

    let mail = MailGenerator.generate(response)

    let message = {
        from : EMAIL,
        to : userEmail,
        subject: "Place Order",
        html: mail
    }

    transporter.sendMail(message).then(() =>{
        return res.status(201).json({
            msg: "you should receive an email",
        })
    }).catch(error => {
        return res.status(500).json({ error })
```

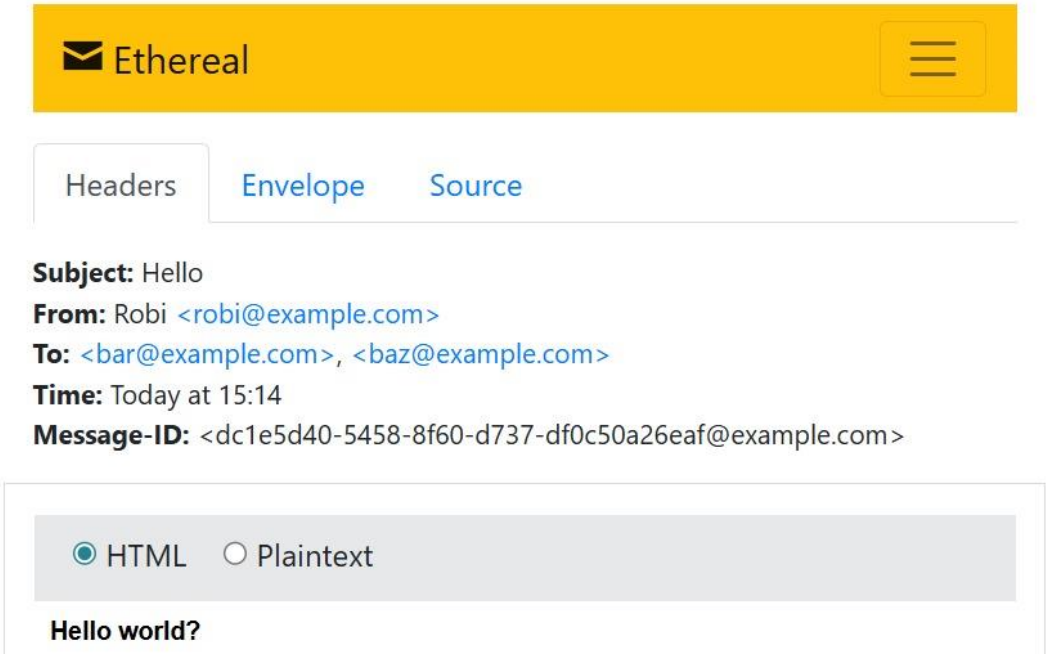
```
    })

    // res.status(201).json("getBill Successfully...!");
}

module.exports = {
    signup,
    getbill
}
```

9. Install extension thunder client
10. Klik new request setelah install extension

    Fitur New Request dalam Thunder Client digunakan untuk mengetes endpoint API, yaitu dengan membuat permintaan POST ke /api/sendmail dan mengirimkan data email melalui body JSON. Dengan fitur ini, proses pengiriman email dapat diuji tanpa harus membuat frontend.
11. Buka terminal dan install nodemailer
12. Coba kirim email ke akun yang belum real



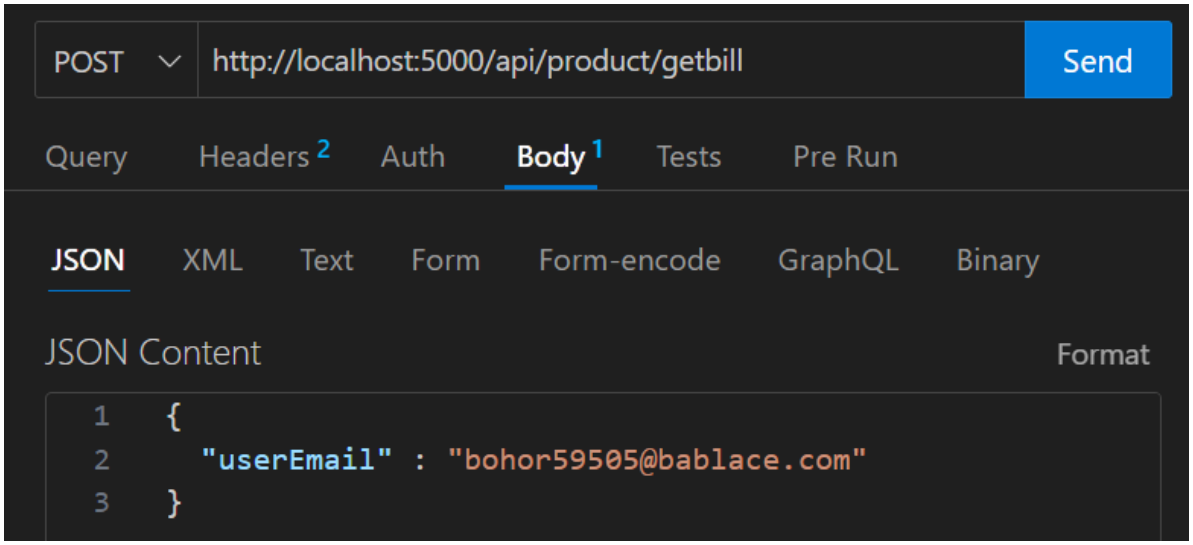
13. Lakukan juga terhadap pengiriman ke email real
14. Buat file env.js

    File env.js digunakan sebagai penyimpan variabel lingkungan penting, seperti email pengirim, app password Gmail, host, dan port SMTP. Tujuannya agar data sensitif tidak ditulis langsung di source code sehingga lebih aman dan mudah dikelola.
15. Masukkan email dan password yang didapat pada apps password google

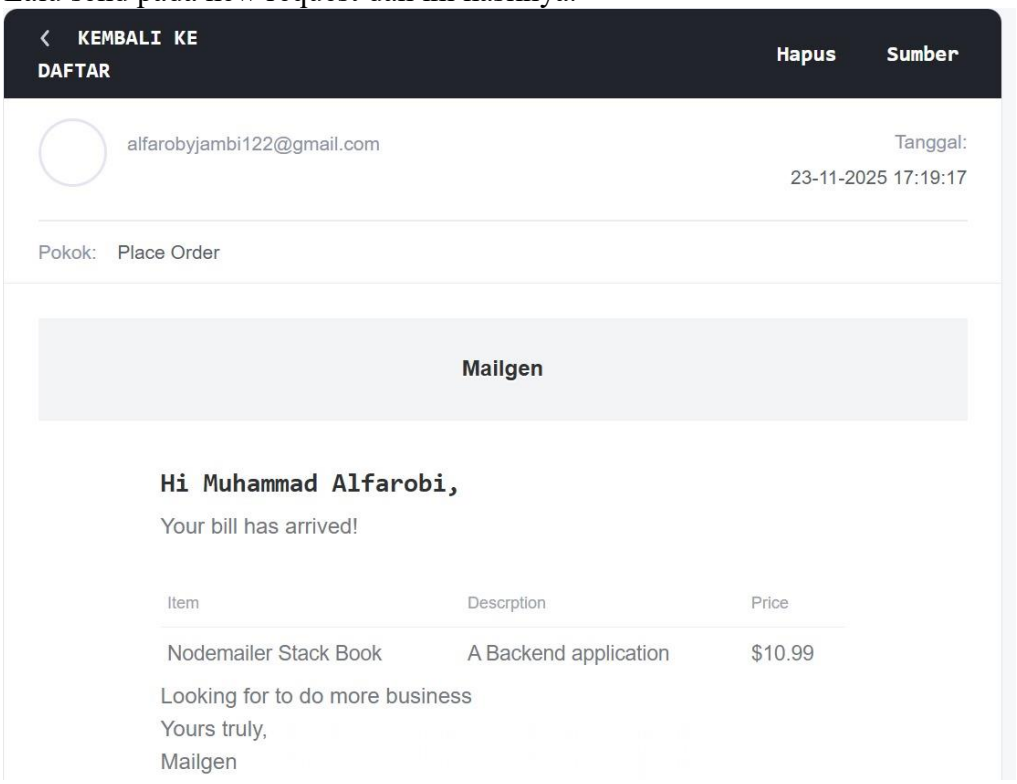
    Source code:

```
module.exports = {
    EMAIL : 'alfarobyjambi122@gmail.com',
    PASSWORD : 'epmyjroatlgracze'
}
```

16. Di new request pada menu body, pakai random mail yang didapat pada web temp_mail

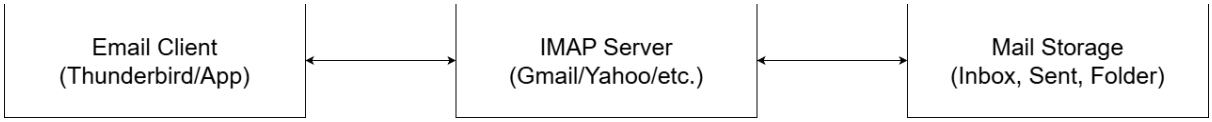


17. Lalu send pada new request dan ini hasilnya.



C. Diagram Terkait Aplikasi

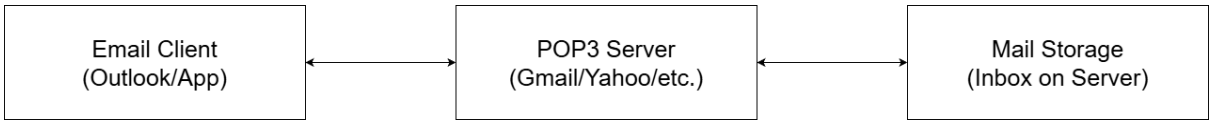
1. IMAP



IMAP (Internet Message Access Protocol) adalah protokol yang digunakan oleh email client untuk membaca email langsung dari server tanpa perlu mengunduh seluruh isi pesan ke perangkat. Prosesnya dimulai ketika email client seperti aplikasi mail di HP atau Thunderbird melakukan autentikasi ke IMAP server menggunakan username dan password (atau token khusus). Setelah berhasil login, client mengirimkan perintah seperti LIST, FETCH, atau SEARCH untuk meminta daftar email atau isi email tertentu.

IMAP server kemudian mengakses penyimpanan email milik pengguna dan mengirimkan kembali metadata seperti subjek, pengirim, atau status pesan, serta isi lengkap email jika diminta. Setiap interaksi seperti membaca, memindahkan, atau menghapus email langsung disinkronkan ke server, sehingga perubahan tersebut muncul di semua perangkat pengguna. Dengan cara kerja ini, IMAP memungkinkan pengguna mengakses email secara real-time pada banyak perangkat tanpa kehilangan data atau melakukan download penuh.

2. POP3



POP3 (Post Office Protocol version 3) adalah protokol yang digunakan untuk mengunduh email dari server ke perangkat pengguna. Berbeda dengan IMAP yang menyimpan email tetap di server, POP3 biasanya memindahkan email ke client sehingga cocok untuk penggunaan single device.

Prosesnya dimulai ketika email client melakukan login ke POP3 server menggunakan username dan password. Setelah autentikasi berhasil, client mengirimkan perintah seperti LIST dan RETR untuk meminta daftar email serta mengunduh seluruh isi pesan. Server kemudian mengirimkan email-email tersebut ke perangkat client. Setelah email berhasil diunduh, client dapat memilih mengirimkan perintah DELE untuk menghapus email dari server, atau membiarkannya tetap tersimpan tergantung pengaturan aplikasi.

3. SMTP

SMTP (Simple Mail Transfer Protocol) adalah protokol standar untuk mengirim email dari pengirim ke server dan dari server ke server lain. Proses dimulai ketika email client baik Node.js, aplikasi mail, maupun webmail membuat pesan email dan mengirimkannya ke SMTP server (misalnya smtp.gmail.com). Server kemudian memeriksa alamat tujuan dan menentukan rute terbaik untuk mengirimkan email tersebut ke mail server penerima.

Setelah menemukan server tujuan, SMTP akan meneruskan pesan melalui koneksi yang aman menggunakan komando standar seperti HELO, MAIL FROM, RCPT TO, dan DATA. Server penerima kemudian menyimpan email tersebut di inbox menggunakan protokol penerimaan seperti IMAP atau POP3. Dengan demikian, SMTP hanya berfokus pada pengiriman email, sedangkan proses pembacaan email diserahkan ke protokol lain seperti IMAP atau POP3.

4. OAuth 2.0

OAuth 2.0 adalah protokol otorisasi yang memungkinkan aplikasi pihak ketiga mengakses data pengguna tanpa harus mengetahui password pengguna. Prosesnya dimulai ketika aplikasi (client) meminta izin kepada authorization server seperti Google atau Facebook. Pengguna kemudian login dan memberikan persetujuan akses. Setelah itu, authorization server mengirimkan authorization code kepada client.

Client menggunakan authorization code tersebut untuk meminta access token. Access token ini berfungsi sebagai “kunci” yang membuktikan bahwa client memiliki izin untuk mengakses data tertentu. Dengan token tersebut, client dapat mengirim permintaan ke resource server misalnya API Gmail, Google Drive, atau data pengguna lainnya. Resource server kemudian memverifikasi token, dan jika valid, ia akan mengirimkan data yang diminta ke aplikasi.

D. Referensi

- bit.ly/tutorial1email
- https://mailtrap.io/blog/imap/
- https://it.telkomuniversity.ac.id/pop3-adalah/
- https://it.telkomuniversity.ac.id/smtp-adalah/
- https://auth0.com/intro-to-iam/what-is-oauth-2